Capítulo 2

Redes Neurais Artificiais

Este capítulo apresenta uma breve introdução sobre os conceitos básicos da teoria de redes neurais artificiais e inicia o estudo do assunto principal deste trabalho – o treinamento de redes do tipo perceptron de múltiplas camadas. O perceptron de múltiplas camadas é a arquitetura de redes neurais artificiais mais utilizada. Sua popularidade é atribuída ao fato de que tem sido aplicada com sucesso a uma grande variedade de problemas de processamento de informação, incluindo classificação de padrões, aproximação de funções e previsão de séries temporais. A derivação do algoritmo de retro-propagação e considerações sobre as virtudes e limitações das redes do tipo perceptron de múltiplas camadas serão brevemente comentadas.

2.1 Introdução

Uma *rede neural artificial* (RNA) é um sistema de processamento de informação que possui algumas características de desempenho em comum com as redes neurais biológicas. Os modelos neurais artificiais têm como principal fonte de inspiração as redes neurais biológicas. A Figura 2.1 apresenta um modelo de um neurônio biológico com a seqüência de propagação dos sinais pela célula.

A natureza das RNA's faz com que seu estudo seja multidisciplinar, envolvendo pesquisadores de diversas áreas, como neurofisiologia, psicologia, física, computação e engenharia.

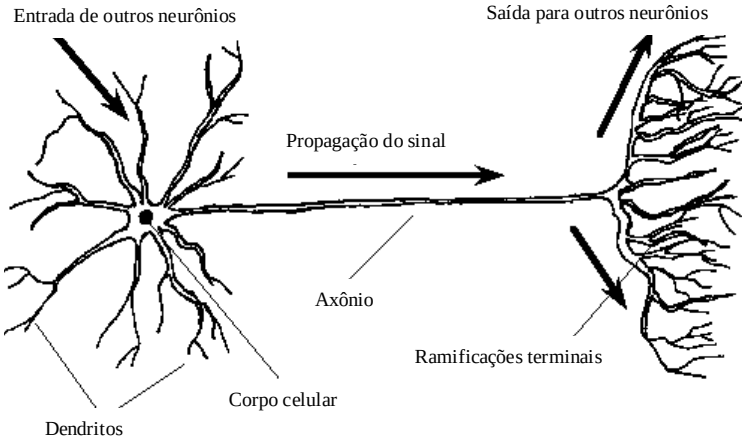


Figura 2.1: Célula neural biológica com a seqüência de propagação do sinal.

Neurofisiologistas e psicólogos estão particularmente interessados em compreender o funcionamento do sistema neural humano. As características de resposta a estímulos apresentada por neurônios individuais, bem como redes de neurônios, são alvo de estudo dos neurofisiologistas, enquanto os psicólogos estudam funções do cérebro no nível cognitivo e estão interessados na utilização de técnicas baseadas em redes neurais para criar modelos detalhados do comportamento humano.

Cientistas da área de computação têm em vista a construção de computadores dotados de processamento paralelo, buscando superar as limitações impostas pelos computadores atuais, que realizam processamento serial em nível simbólico.

Inspirados na habilidade apresentada pelos seres humanos e outros animais no desempenho de funções como o processamento de informações sensoriais e a capacidade de interação com ambientes pouco definidos, os engenheiros estão preocupados em desenvolver sistemas artificiais capazes de desempenhar tarefas semelhantes. Habilidades como capacidade de processamento de informações incompletas ou imprecisas e generalização são propriedades desejadas em tais sistemas.

2.2 Um Breve Histórico

As redes neurais artificiais passaram por um interessante processo de evolução, marcado por um período de grande atividade seguido por anos de estagnação nas pesquisas e pelo ressurgimento do interesse científico como conseqüência do desenvolvimento de novas tecnologias e fundamentos teóricos. A seguir, é apresentado um breve histórico da pesquisa em redes neurais, sendo enfatizados alguns resultados e conceitos considerados relevantes no desenvolvimento deste trabalho (VON ZUBEN, 1993).

Alguns dos mais destacados pesquisadores envolvidos no estudo e aplicação de redes neurais nas últimas três décadas estão relacionados na Tabela 2.1. Esta tabela é dividida de forma a ressaltar o período cronológico mais significativo na atividade científica de cada pesquisador, e é tomada como base no processo de seqüenciamento histórico.

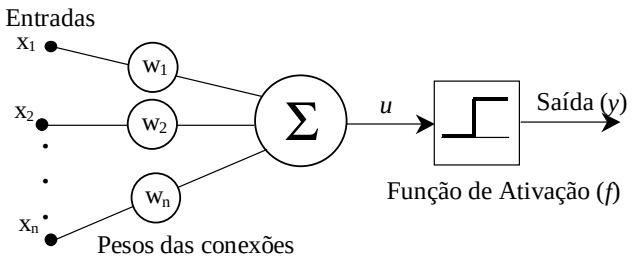


Figura 2.2: Representação funcional de um neurônio.

MCCULLOCH & PITTS (1943) projetaram a estrutura que é conhecida como a primeira rede neural. Estes pesquisadores propuseram um modelo de neurônio como uma unidade de processamento binária (veja Figura 2.2) e provaram que estas unidades são capazes de executar muitas das operações lógicas. Este modelo, apesar de muito simples, trouxe uma grande contribuição para as discussões sobre a construção dos primeiros computadores digitais, permitindo a criação dos primeiros modelos matemáticos de dispositivos artificiais que buscavam analogias biológicas.

Em 1948, N. WIENER (1948) criou a palavra cibernética para descrever, de forma unificada, controle e comunicação nos organismos vivos e nas máquinas.

Em 1949, D. O. HEBB (1949) apresentou uma hipótese a respeito da maneira com que a força das sinapses no cérebro se alteram em resposta à experiência. Em particular ele sugeriu que as conexões entre células que são ativadas ao mesmo tempo tendem a se fortalecer, enquanto que as outras conexões tendem a se enfraquecer. Esta hipótese passou a influir decisivamente na evolução da teoria de aprendizagem em redes neurais artificiais.

Tabela 2.1: História da pesquisa em redes neurais

1943	McCulloch e Pitts
1948	Wiener
1949	Hebb
1957	Rosenblatt
1958	Widrow e Hoff
…	…
1969	Minsky e Papert
…	…
1960-1980	Kohonen, Grossberg, Widrow, Anderson, Caianiello, Fukushima, Ígor Aleksander
…	…
1974	Werbos
…	…
1982	Hopfield
1986	Rumelhart e McClelland

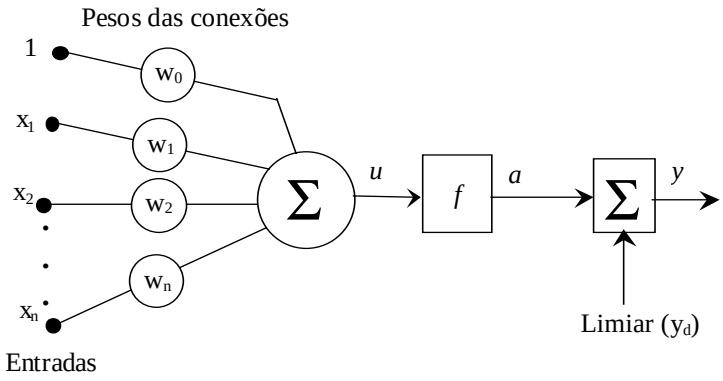


Figura 2.3: O perceptron.

Em 1957, ROSENBLATT (1957) introduziu uma nova abordagem para o problema de reconhecimento de padrões com o desenvolvimento do *perceptron*, cuja representação diagramática é apresentada na Figura 2.3. Rosenblatt também propôs um algoritmo para o ajuste dos pesos do perceptron e provou sua convergência quando os padrões são linearmente separáveis. Por volta do mesmo período, B. WIDROW (1962) e seus colaboradores desenvolveram o *adaline* (***Ada****ptive* ***Lin****ear* ***E****lement*).

Apesar do sucesso do perceptron e do adaline, a pesquisa em redes neurais passou gradualmente a conviver com dois problemas fundamentais. Devido ao fato de a maior parte das pesquisas até então desenvolvidas ser de natureza heurística, o primeiro problema estava vinculado à carência de resultados teóricos que justificassem a manutenção do interesse científico pela área, o que ocasionou a redução na produção de novas idéias. O segundo problema, e talvez o de maior significado histórico, foi a expectativa exagerada criada pelos próprios pesquisadores desta área, não acompanhada de resultados à altura, o que acelerou a queda de financiamentos para pesquisa. Mas foi após a publicação, em 1969, do livro "Perceptrons" de autoria de M. L. MINSKY & S. A. PAPERT (1969), que as pesquisas na área de redes neurais sofreram uma retração significativa. Neste livro, conceitos de matemática moderna como topologia e teoria de grupo são aplicados com o objetivo de analisar as capacidades adaptativas e computacionais de neurônios do tipo apresentado na Figura 2.3. Como resultado, os autores demonstraram que o perceptron, apesar de ser capaz de executar as operações booleanas AND e OR, não é capaz de executar outras operações elementares, como XOR (OU-exclusivo). Além disso, esses autores não acreditavam que uma arquitetura adequada, juntamente com um algoritmo de ajuste de pesos, pudessem ser desenvolvidos de forma a superar esta limitação. Após a publicação destes resultados, a maior parte dos

pesquisadores da área de redes neurais passou a buscar alternativas dentro do campo da engenharia e, principalmente, da lógica matemática, que, na época, encontrava-se em franca expansão devido às grandes conquistas realizadas na área de computação.

Apesar deste êxodo generalizado, um número de pesquisadores continuou a trabalhar com redes neurais nos anos 70. Os nomes de T. Kohonen (Finlândia), S. Grossberg, B. Widrow e J. Anderson (Estados Unidos), E. Caianiello (Itália), K. Fukushima (Japão) e Ígor Aleksander (Inglaterra) estão associados a este período.

Nos anos 80, muitos fatores contribuíram para o ressurgimento definitivo das pesquisas em redes neurais:

- neurofisiologistas foram adquirindo um maior conhecimento sobre o processamento de informações nos organismos vivos;
- avanços tecnológicos tornaram disponível um maior potencial computacional a baixo custo, viabilizando ou facilitando simulações e testes com modelos neurais e
- novas teorias para a implementação de algoritmos adaptativos foram desenvolvidas, permitindo a aplicação em sistemas reais.

O ganhador do Prêmio Nobel J. HOPFIELD (1982), do Instituto de Tecnologia da Califórnia, juntamente com D. TANK, um pesquisador da AT&T, desenvolveram grande quantidade de modelos de redes neurais baseadas em pesos fixos e ativações adaptativas. Estas redes podem servir como memórias autoassociativas e serem usadas para resolver problemas de otimização restrita como o caso do "Caixeiro Viajante" (URL 3). A rede de Hopfield pode ser considerada como um sistema dinâmico com um número finito de estados de equilíbrio, de forma que o sistema invariavelmente irá evoluir para um destes estados ou para uma seqüência periódica de estados a partir de uma condição inicial. É também natural que a localização destes estados de equilíbrio possa ser controlada pela intensidade das conexões (pesos) da rede neural.

A conclusão interessante adotada por Hopfield foi que tais estados de equilíbrio podem ser utilizados como dispositivos de memória. De forma distinta daquela utilizada pelos computadores convencionais, em que o acesso à informação armazenada se dá por meio de um endereço, o acesso ao conteúdo da memória de uma rede de Hopfield se dá permitindo que a rede evolua com o tempo para um de seus estados de equilíbrio. Tais modelos de memória são denominados memórias endereçáveis por conteúdo.

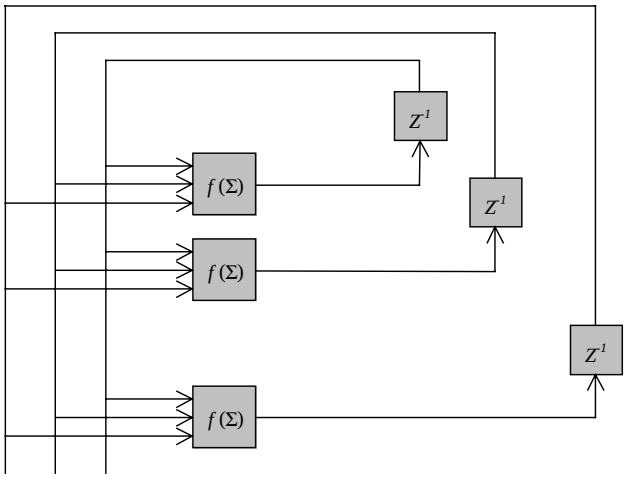


Figura 2.4: Rede Neural de Hopfield.

O trabalho de Hopfield com este tipo de rede simétrica recorrente atraiu, principalmente, matemáticos e engenheiros para a pesquisa nesta área, e as redes de Hopfield foram estudadas como memórias distribuídas e também utilizadas como ferramentas na solução de problemas de otimização restrita.

No entanto, o fato que efetivamente colocou a área de redes neurais como uma das prioritárias na obtenção de recursos foi o desenvolvimento de um método para ajuste de parâmetros de redes não-recorrentes de múltiplas camadas. Este método, baseado em um algoritmo denominado retro-propagação (*backpropagation*), tornou-se largamente conhecido após a publicação, em 1986, do livro *Parallel Distributed Processing*, editado por J. L. MCCLELLAND & D. E. RUMELHART (1986a-1986b), fazendo com que pesquisadores das mais diferentes áreas passassem a visualizar interessantes aplicações para redes neurais artificiais. A importância deste método justifica um tratamento mais aprofundado, a ser desenvolvido nas próximas seções.

2.3 Características Principais

As redes neurais artificiais têm sido desenvolvidas como generalizações de modelos matemáticos da cognição humana ou biologia neural, assumindo que:

- o processamento da informação ocorre em vários elementos chamados *neurônios*;
- os sinais são propagados de um elemento a outro através de *conexões*;
- cada conexão possui um *peso* associado, que, em uma rede neural típica, pondera o sinal transmitido; e

- cada neurônio (ou unidade) aplica uma *função de ativação* (geralmente não-linear) à sua entrada de rede (soma ponderada dos sinais de entrada) para determinar sua saída.

Uma rede neural pode ser caracterizada por três aspectos principais: (1) o padrão de conexões entre as unidades (*arquitetura*), (2) o método de determinação dos pesos das conexões (*algoritmo de treinamento ou aprendizado*) e (3) sua *função de ativação*.

Os modelos neurais artificiais oferecem um paradigma atrativo, pois "aprendem" a resolver problemas através de exemplos.

O treinamento de RNA's pode ser dividido em:

- supervisionado: necessita de um "professor" durante a fase de aprendizagem, que antecede a utilização (execução) da rede; e
- não-supervisionado: direcionado por correlações existentes nos dados de entrada e, portanto, não necessita de um "professor".

Existem vários tipos de modelos de RNA's atualmente. Novos modelos (ou pelo menos variações de alguns já existentes) são propostos constantemente. A seguir apresentamos uma lista com algumas das arquiteturas mais conhecidas até hoje. A distinção principal entre os modelos citados refere-se ao tipo de treinamento (URL 1).

Treinamento não-supervisionado:

1) Redes recorrentes:

- Grossberg Aditivo (AG)
- *Adaptive Resonance Theory* (ART1)
- Hopfield Simétrico e Assimétrico (DH/CH)
- Memória Associativa Bidirecional (BAM)
- Memória Associativa Temporal (TAM)
- Mapa Auto-organizável de Kohonen (SOM)
- Aprendizado Competitivo

2) Redes somente com propagação positiva (*feedforward*):

- *Learning Matrix* (LM)
- *Driver-Reinforcement Learning* (DR)
- Memória Associativa Linear (LAM)
- *Counterprogation* (CPN)

Treinamento Supervisionado:

1) Redes Recorrentes:

- Máquina de Boltzmann (BM)
- *Mean Field Annealing* (MFA)
- *Cascade Correlation* Recorrente (RCC)
- Aprendizado Recorrente em Tempo Real (RTRL)
- Filtro de Kalman Recorrente (EKF)

2) Redes somente com propagação positiva (*feedforward*):

- *Perceptron*
- *Adaline, Madaline*
- Retro-propagação – *Backpropagation* (BP)
- Máquina de Cauchy (CM)
- *Artmap*
- Rede Lógica Adaptativa (ALN)
- *Cascade Correlation* (CasCor)
- Filtro de Kalman (EKF)
- *Learning Vector Quantization* (LVQ)
- Rede Neural Probabilística (PNN)

2.4 Redes do Tipo Perceptron de Múltiplas Camadas (*MLP*)

As arquiteturas do tipo perceptron de múltiplas camadas (MLP) constituem os modelos neurais artificiais mais utilizados e conhecidos.

Tipicamente, esta arquitetura consiste de um conjunto de unidades sensoriais que formam uma camada de entrada, uma ou mais camadas intermediárias (ou escondidas) de unidades computacionais e uma camada de saída. Os sinais de entrada são propagados camada a camada pela rede em uma direção positiva, ou seja, da entrada para a saída. Esta arquitetura representa uma generalização do perceptron apresentado anteriormente.

As redes do tipo MLP tem sido utilizadas com sucesso para a solução de vários problemas envolvendo altos graus de não-linearidade. Seu treinamento é do tipo supervisionado e utiliza um algoritmo muito popular chamado retro-propagação do erro (*error backpropagation*). Este algoritmo é baseado numa regra de aprendizagem que "corrige" o erro durante o treinamento (HAYKIN, 1994).

Basicamente, o processo de retro-propagação do erro é constituído de duas fases: uma fase de propagação do sinal funcional (*feedforward*) e uma de retro-propagação do erro (*backpropagation*). Na fase positiva, os vetores de dados são aplicados às unidades de entrada, e seu efeito se propaga pela rede, camada a camada. Finalmente, um conjunto de saídas é produzido como resposta da rede. Durante a fase positiva, os pesos das conexões são mantidos fixos. Na retro-propagação do erro, por outro lado, os pesos são ajustados de acordo com uma regra de correção do erro. Especificamente, a resposta da rede em um instante de tempo é subtraída da saída desejada (*target*) para produzir um *sinal de erro*. Este sinal de erro é propagado da saída para a entrada, camada a camada, originando o nome "retro-propagação do erro". Os pesos são ajustados de forma que a distância entre a resposta da rede e a resposta desejada seja reduzida.

Uma rede do tipo MLP possui três características distintas:

- função de ativação;
- número de camadas e unidades intermediárias e
- forma das conexões.

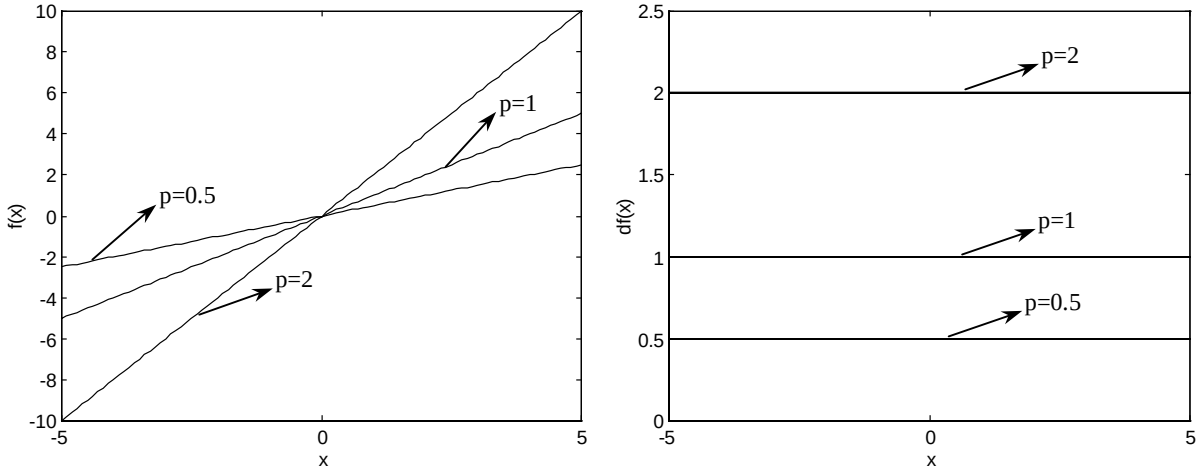
2.4.1 Função de Ativação

O modelo de cada unidade da rede pode incluir uma *não-linearidade* na sua saída. É importante enfatizar que a não-linearidade deve ser suave, ou seja, diferenciável, diferentemente da função sinal utilizada pelo perceptron original.

A função de ativação representa o efeito que a entrada interna e o estado atual de ativação exercem na definição do próximo estado de ativação da unidade. Quando propriedades dinâmicas estão envolvidas na definição do estado de ativação, equações diferenciais (caso contínuo) ou a diferenças (caso discreto) são empregadas. Tendo em vista a simplicidade desejada para as unidades processadoras, geralmente define-se seu estado de ativação como uma função algébrica da entrada interna atual, independente de valores passados do estado de ativação ou mesmo da entrada interna. Geralmente, esta função é monotonicamente não-decrescente e apresenta um tipo de não-linearidade associada ao efeito da saturação. A seguir são descritos alguns tipos de função de ativação empregados na literatura para as arquiteturas do tipo MLP (KOSKO, 1992; HAYKIN, 1994).

Função Linear

Este tipo de função de ativação é muito utilizado nas unidades que compõem a camada de saída das arquiteturas MLP.



(a) (b)

Figura 2.5: (a) Função linear. (b) Sua derivada em relação a entrada interna.

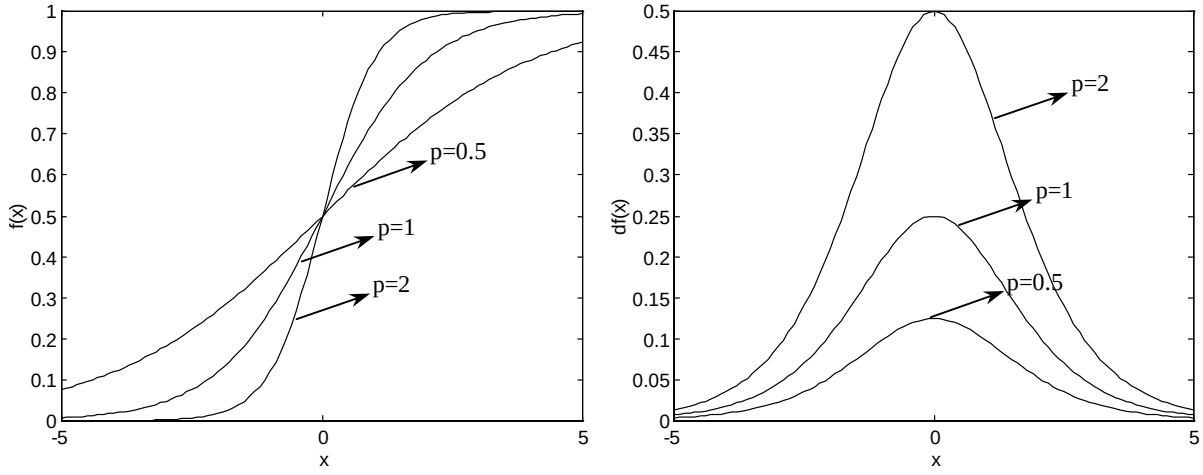
A saída linear com $p = 1$ simplesmente repete o sinal que entra no neurônio na sua saída. A Figura 2.5 apresenta saídas lineares e suas derivadas.

A expressão para esta função de ativação e sua derivada é:

$$f(x) = p.x, \qquad f'(x) = p$$

Função Logística

A Figura 2.6 (a) mostra que a função logística possui intervalo de variação entre 0 e 1.



(a) (b)

Figura 2.6: (a) Função logística. (b) Sua derivada em relação a entrada interna.

A origem deste tipo de função está vinculada à preocupação em limitar o intervalo de variação da derivada da função, pela inclusão de um efeito de saturação. Sua derivada também é uma função contínua (ver Figura 2.6).

$$f(x) = \frac{e^{px}}{1+e^{px}} = \frac{1}{1+e^{-px}}, \qquad f'(x) = p\, f(x).\left(1 - f(x)\right)$$

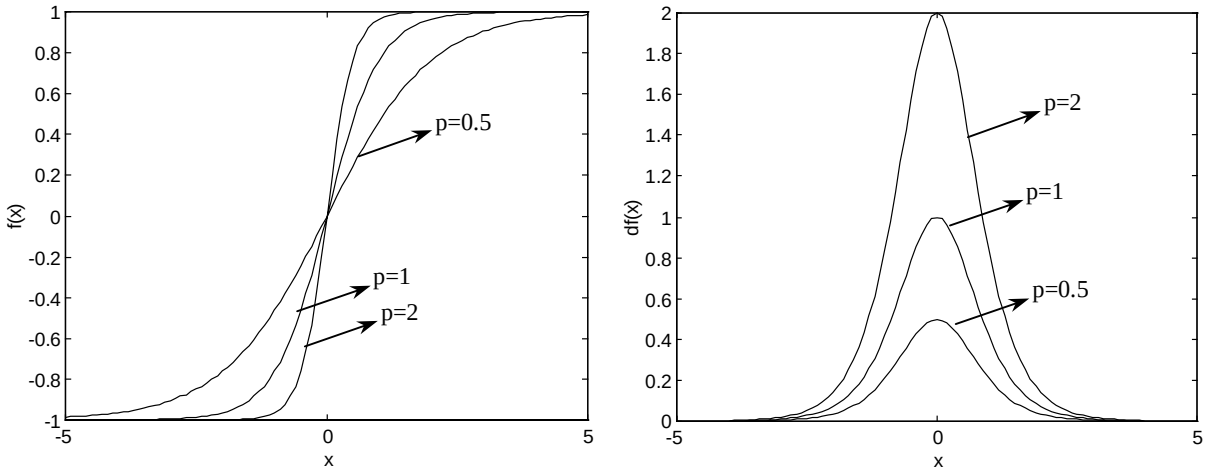
Função Tangente Hiperbólica

Pelo fato da função logística apresentar valores de ativação apenas no intervalo (0, 1), em muitos casos ela é substituída pela função tangente hiperbólica, que preserva a forma sigmoidal da função logística, mas assume valores positivos e negativos ($f(x) \in (-1, 1)$). A função tangente hiperbólica e sua derivada (ver Figura 2.7) são dadas pelas expressões:

$$f(x) = \frac{e^{px} - e^{-px}}{e^{px} + e^{-px}} = \tanh(px), \qquad f'(x) = p\left(1 - f(x)^2\right)$$

Em todas as redes implementadas e testadas neste trabalho utilizou-se função de ativação linear na saída da rede. A função tangente hiperbólica foi a escolhida para as unidades intermediárias das redes do tipo MLP fixas. No Capítulo 7, são feitos estudos sobre modelos construtivos de redes, onde alguns destes modelos utilizam funções de ativação diferentes da tangente hiperbólica na camada intermediária. Estes modelos serão especificados nas seções correspondentes à sua descrição.

A função tangente hiperbólica pode ser transladada para o intervalo (0, 1), assim como a função logística pode ser transladada para o intervalo (-1, 1).



(a) (b)

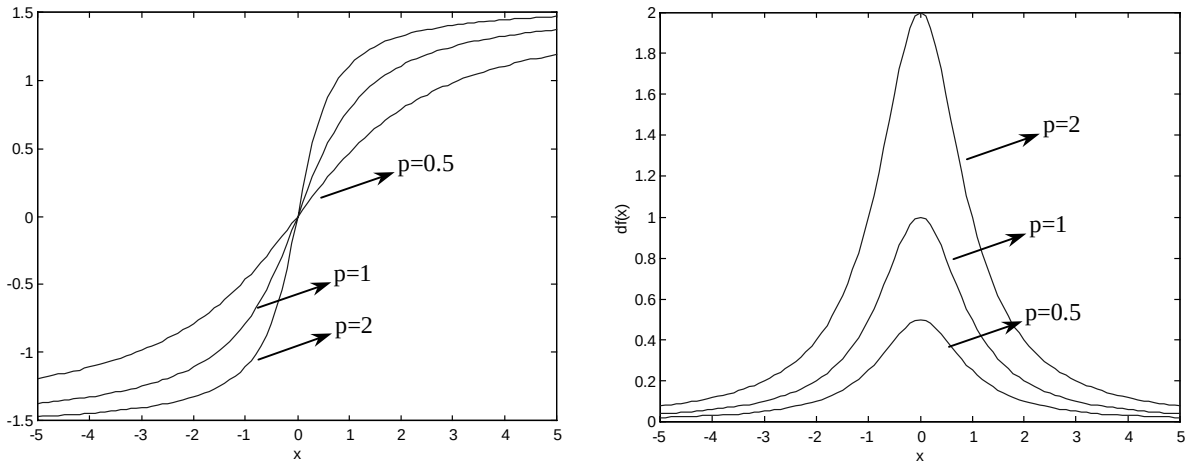
Figura 2.7: (a) Função tangente hiperbólica. (b) Sua derivada em relação a entrada interna.

Função Arco-Tangente

Esta função possui valores de ativação no intervalo (-π/2, π/2), e pode ser apresentada como uma alternativa à função tangente hiperbólica para a implementação computacional, pois requer menos cálculos para sua elaboração. Comparações sucintas entre o tempo de processamento das funções tanh(.), atan(.), e logística mostram que a função atan(.) possui o menor tempo de processamento, seguida da função logística e por último a função tangente hiperbólica.

A função arco tangente é dada pela expressão abaixo:

$$f(x) = \text{atan}(px) \qquad f'(x) = p.\frac{1}{1+x^2}$$



(a) (b)

Figura 2.8: (a) Função arco-tangente (atan). (b) Sua derivada em relação a entrada interna.

Os limites dos intervalos da função apresentada acima podem ser transladados para o intervalo (-1,1) comumente utilizado.

2.4.2 Noções Gerais

A Figura 2.9 apresenta uma arquitetura do tipo MLP com duas camadas intermediárias. A rede apresentada aqui possui todas as conexões, o que significa que um neurônio em qualquer camada da rede está conectado a todas as outras unidades (neurônios) na camada anterior. O fluxo de sinais através da rede é feito positivamente, da esquerda para a direita, camada a camada.

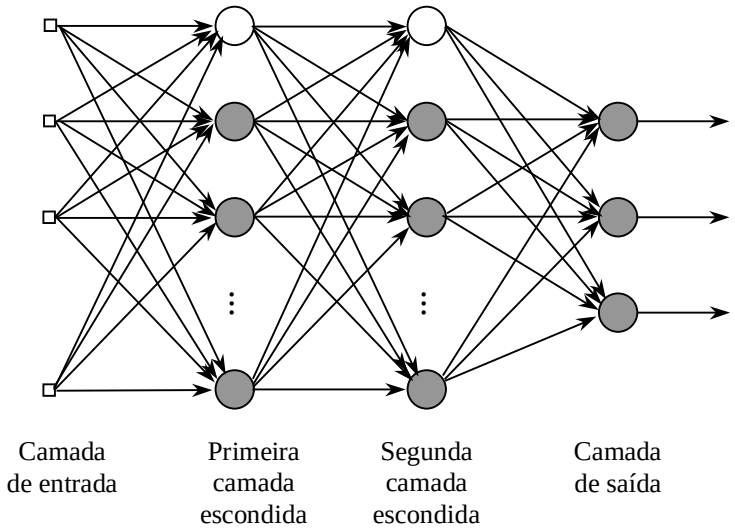


Figure 2.9: Arquitetura MLP com duas camadas intermediárias.

A Figura 2.10 mostra apenas uma parte da rede. Nesta rede, dois tipos de sinais podem ser identificados:

- *sinal funcional*: um sinal funcional é um sinal de entrada (estímulo) que chega na entrada e é propagado positivamente (neurônio a neurônio) através da rede, e aparece na saída como um sinal de saída.
- *sinal de erro*: os sinais de erro originam-se nas saídas e são retro-propagados (neurônio a neurônio) através da rede.

A camada de entrada geralmente é composta por neurônios sensoriais, ou seja, unidades que não modificam os sinais externos, apenas os distribuem para a primeira camada intermediária. As unidades de saída constituem a camada de saída da rede, e as demais unidades constituem as camadas intermediárias. As camadas intermediárias são todas aquelas que não fazem parte da entrada e nem da saída.

Cada unidade intermediária ou de saída é responsável por duas tarefas:

- calcular o sinal na saída da unidade, que geralmente é expresso como uma função não-linear do sinal de entrada e pesos sinápticos associados e
- calcular uma estimativa instantânea do vetor gradiente, que é necessário para a retro-propagação do erro através da rede.

Para facilitar a derivação da regra de retro-propagação, primeiramente apresentamos um resumo da notação utilizada.

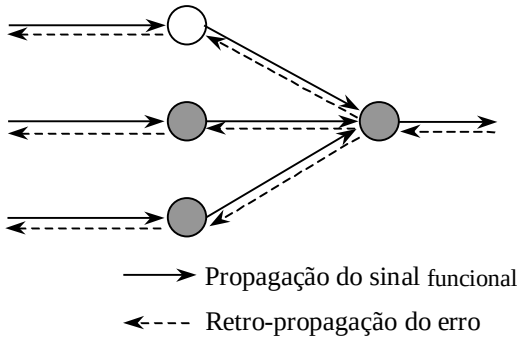


Figura 2.10: Ilustração das direções de propagação do sinal funcional e do erro.

Notação

i, j	índices referentes a diferentes neurônios da rede
n	n-ésimo vetor de entrada (iteração)
N	número de amostras (padrões de treinamento)
M	número de camadas
$y_j(n)$	sinal de saída da unidade j na iteração n
$e_j(n)$	sinal de erro da unidade de saída j na iteração n
$w_{i,j}(n)$	peso sináptico conectando a saída da unidade i à entrada da unidade j na iteração n
$u_j(n)$	ativação da unidade j na iteração n; sinal a ser aplicado à não-linearidade
$f_j(.)$	função de ativação associada à unidade j
$\mathbf{X}$	matriz de dados de entrada (amostras de treinamento)
$\mathbf{S}$	matriz de dados de saída (saídas desejadas)
$x_i(n)$	i-ésimo elemento do vetor de entradas
$s_j(n)$	j-ésimo elemento do vetor de saídas
α	taxa de aprendizagem

Todas as letras minúsculas em **negrito** (**a**, **b**, **c**) representam vetores, as letras maiúsculas em **negrito** (**A**, **B**, **C**) denotam matrizes e as letras em *itálico* (*a*, *b*, *c*) representam escalares.

As matrizes $\mathbf{W}^m$ (para $m = 0, 1, \ldots, M - 1$; onde M é o número de camadas da rede) possuem dimensão $S^{m+1} \times S^m$, onde S^0 = número de entradas da rede; e os vetores $\mathbf{b}^m$ possuem dimensão $S^{m+1} \times 1$.

2.4.3 Derivação do Algoritmo de Retro-Propagação (*Backpropagation*)

O algoritmo a ser apresentado representa uma forma eficiente e elegante de implementação computacional da regra da cadeia, principalmente quando a ferramenta utilizada possui capacidade de processamento matricial, como os pacotes Matlab® e Mathematica®.

Para simplificar o desenvolvimento do algoritmo de retro-propagação, utilizaremos a notação abreviada para uma arquitetura genérica com três camadas (ver Figura 2.11) (DEMUTH & BEALE, 1993; HAGAN *et. al.*, 1997).

Para redes de múltiplas camadas, a saída de uma camada torna-se a entrada da camada seguinte. A equação que descreve esta operação é:

$$\mathbf{u}^{m+1} = \mathbf{f}^{m+1}(\mathbf{W}^{m+1}\mathbf{y}^{m} + \mathbf{b}^{m+1}), \text{ para } m = 0, 1, \ldots, M-1, \tag{2.1}$$

onde M é o número de camadas da rede. As unidades na primeira camada recebem entradas externas agrupadas em um vetor na forma:

$$\mathbf{y}^{0} = \mathbf{x}, \tag{2.2}$$

que representa a condição inicial para a equação (2.1). As saídas das unidades localizadas na última camada são consideradas as saídas da rede:

$$\mathbf{y} = \mathbf{y}^{M}. \tag{2.3}$$

Pela Figura 2.11 percebemos que a saída da rede pode ser expressa apenas em função do vetor de entradas $\mathbf{x}$, das matrizes de pesos $\mathbf{W}^m$ e dos vetores de limiares $\mathbf{b}^m$, cuja expressão é:

$$\mathbf{y}^3 = \mathbf{f}^3(\mathbf{W}^3\mathbf{f}^2(\mathbf{W}^2\mathbf{f}^1(\mathbf{W}^1\mathbf{x} + \mathbf{b}^1)+\mathbf{b}^2)+\mathbf{b}^3). \tag{2.4}$$

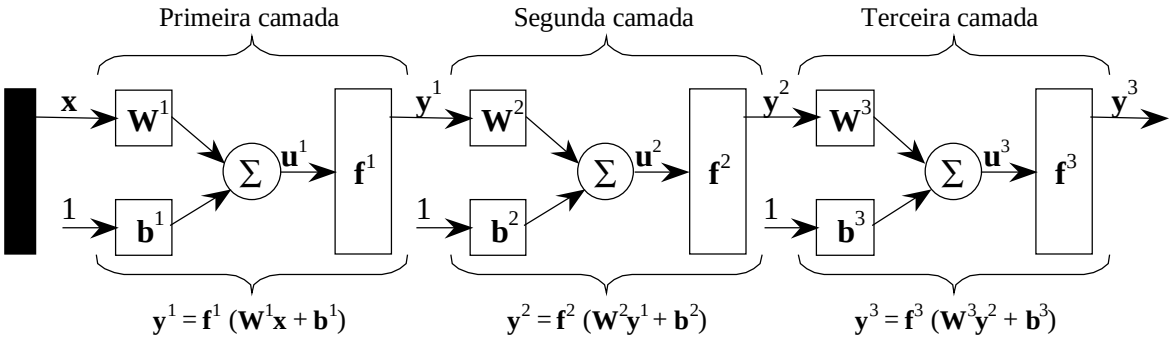


Figura 2.11: Rede com três camadas. Notação abreviada.

2.4.3.1 Índice de Desempenho

O algoritmo de retro-propagação para as redes de múltiplas camadas é uma generalização do *método dos quadrados mínimos* (LS – do inglês *Least Squares*) e utiliza como medida de desempenho o *erro quadrático médio* (MSE – do inglês *mean squared error*). Inicialmente, é apresentado um conjunto de exemplos:

$$\{(\mathbf{x}_1, \mathbf{s}_1), (\mathbf{x}_2, \mathbf{s}_2), \ldots, (\mathbf{x}_N, \mathbf{s}_N)\}, \tag{2.5}$$

onde $\mathbf{x}_n$ é a n-ésima entrada para a rede e $\mathbf{s}_n$ a saída desejada correspondente ($n = 1, \ldots, N$). Das equações (2.3) e (2.4) vemos que, se θ é o vetor de parâmetros da rede (pesos e limiares), então o vetor de saídas da rede pode ser dado na forma:

$$y^M = y(\theta, x)$$

Após cada entrada ser aplicada à rede, a saída produzida pela rede é comparada com a saída desejada. O algoritmo deve ajustar os parâmetros da rede (pesos e limiares), com o objetivo de minimizar a esperança matemática do erro quadrático médio:

$$J(\theta) = E(e(\theta)^2) = E((s - y(\theta))^2), \tag{2.6}$$

Se a rede possui múltiplas saídas, a equação (2.6) pode ser generalizada para

$$J(\theta) = E(\mathbf{e}(\theta)^T \mathbf{e}(\theta)) = E((\mathbf{s} - \mathbf{y}(\theta))^T (\mathbf{s} - \mathbf{y}(\theta))). \tag{2.7}$$

Como no algoritmo LS, o erro quadrático médio será aproximado por

$$\hat{J}(\theta) = \mathbf{e}(n)^T \mathbf{e}(n) = (\mathbf{s}(n) - \mathbf{y}(n))^T (\mathbf{s}(n) - \mathbf{y}(n)), \tag{2.8}$$

onde a esperança do erro quadrático foi substituída pelo erro na iteração n. Para não sobrecarregar a notação empregada, consideraremos $\hat{J}(\theta) = J(\theta)$.

A lei de ajuste conhecida como *steepest descent* para minimizar o erro quadrático é dada por:

$$w_{i,j}^m(n+1) = w_{i,j}^m(n) - \alpha \frac{\partial J(n)}{\partial w_{i,j}^m}, \tag{2.9}$$

$$b_i^m(n+1) = b_i^m(n) - \alpha \frac{\partial J(n)}{\partial b_i^m}, \tag{2.10}$$

onde α é a taxa de aprendizagem.

A parte mais elaborada deste cálculo está na determinação das derivadas parciais que irão gerar os componentes do vetor gradiente.

2.4.3.2 Regra da Cadeia

Para uma rede com múltiplas camadas o erro não é função direta dos pesos nas camadas intermediárias, por isso o cálculo destas derivadas não é imediato.

Como o erro é função indireta dos pesos nas camadas intermediárias, a regra da cadeia deverá ser usada para o cálculo das derivadas. O conceito da regra da cadeia será utilizado na determinação das derivadas das equações (2.9) e (2.10):

$$\frac{\partial J}{\partial w_{i,j}^m} = \frac{\partial J}{\partial u_i^m} \times \frac{\partial u_i^m}{\partial w_{i,j}^m}, \tag{2.11}$$

$$\frac{\partial J}{\partial b_i^m} = \frac{\partial J}{\partial u_i^m} \times \frac{\partial u_i^m}{\partial b_i^m}. \tag{2.12}$$

O segundo termo de cada uma das equações acima pode ser facilmente calculado, uma vez que a ativação da camada *m* é uma função explícita dos pesos e limiares nesta camada:

$$u_i^m = \sum_{j=1}^{S^{m-1}} w_{i,j}^m y_j^{m-1} + b_i^m. \tag{2.13}$$

Entretanto

$$\frac{\partial u_i^m}{\partial w_{i,j}^m} = y_j^{m-1}, \frac{\partial u_i^m}{\partial b_i^m} = 1. \tag{2.14}$$

Definindo agora a *sensibilidade* de *J* a mudanças no *i*-ésimo elemento da ativação da rede na camada *m* como:

$$\delta_i^m \equiv \frac{\partial J}{\partial u_i^m}, \tag{2.15}$$

então as equações (2.11) e (2.12) podem ser simplificadas por

$$\frac{\partial J}{\partial w_{i,j}^m} = \delta_i^m y_j^{m-1}, \tag{2.16}$$

$$\frac{\partial J}{\partial b_i^m} = \delta_i^m. \tag{2.17}$$

Agora é possível aproximar as equações (2.9) e (2.10) por

$$w_{i,j}^m(n+1) = w_{i,j}^m(n) - \alpha \delta_i^m y_j^{m-1}, \tag{2.18}$$

$$b_i^m(n+1) = b_i^m(n) - \alpha \delta_i^m. \tag{2.19}$$

Em notação matricial, as equações acima tornam-se:

$$\mathbf{W}^m(n+1) = \mathbf{W}^m(n) - \alpha \boldsymbol{\delta}^m (\mathbf{y}^{m-1})^T, \tag{2.20}$$

$$\mathbf{b}^m(n+1) = \mathbf{b}^m(n) - \alpha\boldsymbol{\delta}^m, \tag{2.21}$$

onde

$$\boldsymbol{\delta}^m \equiv \frac{\partial J}{\partial \mathbf{u}^m} = \begin{bmatrix} \dfrac{\partial J}{\partial u_1^m} \\ \dfrac{\partial J}{\partial u_2^m} \\ \vdots \\ \dfrac{\partial J}{\partial u_{S^m}^m} \end{bmatrix}. \tag{2.22}$$

2.4.3.3 Retro-propagação das Sensibilidades

Ainda é necessário calcular as sensibilidades δ^m, que requer outra aplicação da regra da cadeia. É este processo que dá origem ao termo *retro-propagação*, pois descreve a relação de recorrência na qual a sensibilidade na camada *m* é calculada a partir da sensibilidade na camada *m* + 1.

Para derivar a relação de recorrência das sensibilidades, utilizaremos a seguinte matriz jacobiana:

$$\frac{\partial \mathbf{u}^{m+1}}{\partial \mathbf{u}^m} = \begin{bmatrix} \dfrac{\partial u_1^{m+1}}{\partial u_1^m} & \dfrac{\partial u_1^{m+1}}{\partial u_2^m} & \cdots & \dfrac{\partial u_1^{m+1}}{\partial u_{S^m}^m} \\ \dfrac{\partial u_2^{m+1}}{\partial u_1^m} & \dfrac{\partial u_2^{m+1}}{\partial u_2^m} & \cdots & \dfrac{\partial u_2^{m+1}}{\partial u_{S^m}^m} \\ \vdots & \vdots & \ddots & \vdots \\ \dfrac{\partial u_{S^{m+1}}^{m+1}}{\partial u_1^m} & \dfrac{\partial u_{S^{m+1}}^{m+1}}{\partial u_2^m} & \cdots & \dfrac{\partial u_{S^{m+1}}^{m+1}}{\partial u_{S^m}^m} \end{bmatrix}. \tag{2.23}$$

Em seguida desejamos encontrar uma expressão para esta matriz. Considere o elemento *i*, *j* da matriz:

$$\frac{\partial u_i^{m+1}}{\partial u_j^m} = w_{i,j}^{m+1} \frac{\partial y_j^m}{\partial u_j^m} = w_{i,j}^{m+1} \frac{\partial f^m(u_j^m)}{\partial u_j^m} = w_{i,j}^{m+1} \dot{f}^m(u_j^m), \tag{2.24}$$

onde

$$\dot{f}^m(u_j^m) = \frac{\partial f^m(u_j^m)}{\partial u_j^m}. \tag{2.25}$$

Entretanto a matriz jacobiana pode ser escrita

$$\frac{\partial \mathbf{u}^{m+1}}{\partial \mathbf{u}^m} = \mathbf{W}^{m+1}\dot{\mathbf{F}}^m(\mathbf{u}^m), \tag{2.26}$$

onde

$$\dot{\mathbf{F}}^m(\mathbf{u}^m) = \begin{bmatrix} \dot{f}^m(u_1^m) & 0 & \cdots & 0 \\ 0 & \dot{f}^m(u_2^m) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \dot{f}^m(u_{S^m}^m) \end{bmatrix}. \tag{2.27}$$

Agora podemos escrever a relação de recorrência para a sensibilidade utilizando a regra da cadeia em forma matricial:

$$\begin{aligned} \boldsymbol{\delta}^m &= \frac{\partial J}{\partial \mathbf{u}^m} = \left(\frac{\partial \mathbf{u}^{m+1}}{\partial \mathbf{u}^m} \right)^T \frac{\partial J}{\partial \mathbf{u}^{m+1}} = \dot{\mathbf{F}}^m(\mathbf{u}^m)(\mathbf{W}^{m+1})^T \frac{\partial J}{\partial \mathbf{u}^{m+1}} \\ &= \dot{\mathbf{F}}^m(\mathbf{u}^m)(\mathbf{W}^{m+1})^T \boldsymbol{\delta}^{m+1}. \end{aligned} \tag{2.28}$$

Observe que as sensibilidades são propagadas da última para a primeira camada através da rede:

$$\delta^M \rightarrow \delta^{M-1} \rightarrow \ldots \rightarrow \delta^2 \rightarrow \delta^1 \tag{2.29}$$

Ainda existe um último passo a ser executado para que o algoritmo de retro-propagação fique completo. Precisamos do ponto de partida, δ^M, para a relação de recorrência da equação (2.28). Este ponto é obtido na última camada:

$$\delta_i^M = \frac{\partial J}{\partial u_i^M} = \frac{\partial (\mathbf{s}-\mathbf{y})^T(\mathbf{s}-\mathbf{y})}{\partial u_i^M} = \frac{\partial \sum_{j=1}^{S^M} (s_j - y_j)^2}{\partial u_i^M} = -2(s_i - y_i)\frac{\partial y_i}{\partial u_i^M}. \tag{2.30}$$

Como

$$\frac{\partial y_i}{\partial u_i^M} = \frac{\partial y_i^M}{\partial u_i^M} = \frac{\partial f^M(u_j^M)}{\partial u_i^M} = \dot{f}^M(u_j^M), \tag{2.31}$$

podemos escrever

$$\delta_i^M = -2(s_i - y_i)\dot{f}^M(u_j^M). \tag{2.32}$$

A expressão acima pode ser colocada em forma matricial, resultando

$$\boldsymbol{\delta}^M = -2\dot{\mathbf{F}}^M(\mathbf{u}^M)(\mathbf{s}-\mathbf{y}). \tag{2.33}$$

A Figura 2.12 mostra uma adaptação (utilizando notação matricial abreviada) do resultado de NARENDRA & PARTHASARATHY (1990) e descreve um esquema gráfico de fluxo de sinais para o algoritmo de treinamento de retro-propagação, sendo bastante útil para descrever todo o equacionamento do algoritmo de retro-propagação.

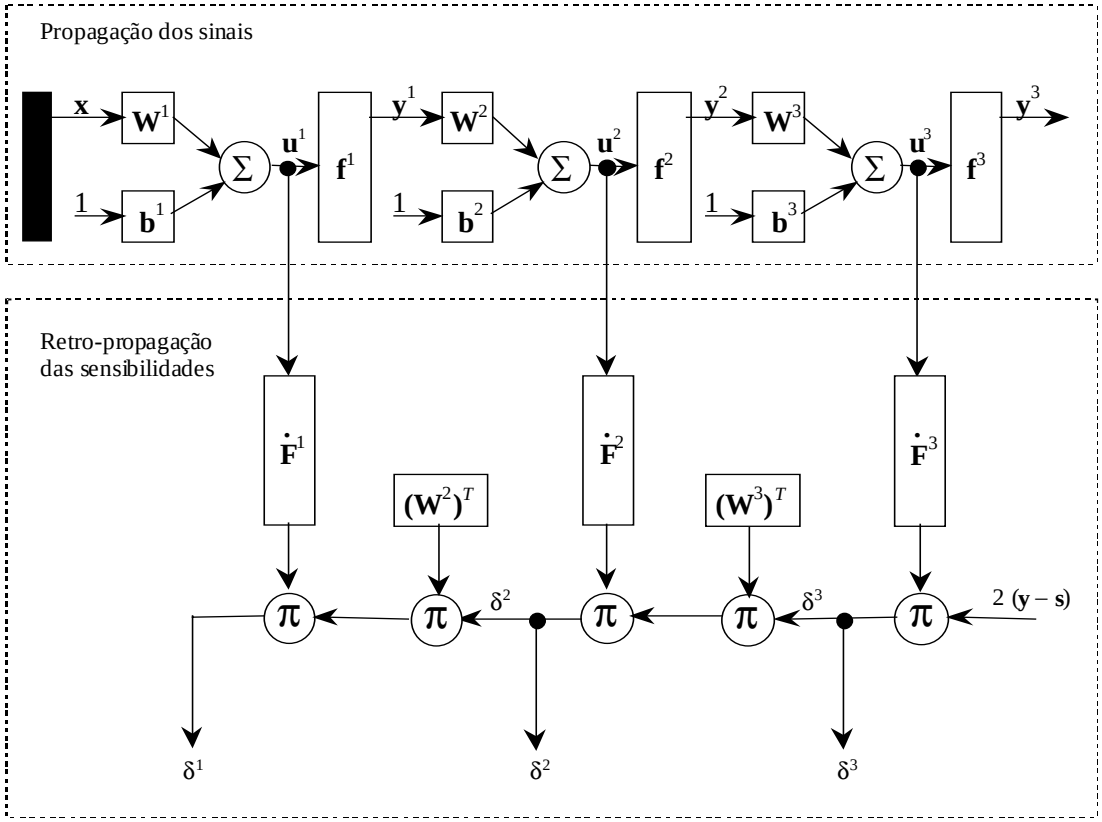


Figura 2.12: Gráfico arquitetural de uma rede com três camadas, representando a fase de propagação dos sinais e retro-propagação das sensibilidades.

2.4.4 Treinamento Local ou em Lote

Em aplicações práticas do algoritmo de retro-propagação, o aprendizado é resultado de apresentações repetidas de todas as amostras do conjunto de treinamento ao MLP. Cada apresentação de todo o conjunto de treinamento durante o processo de aprendizagem é chamada de *época*. O processo de aprendizagem é repetido época após época, até que o conjunto de pesos e limiares estabilize e o erro quadrático médio do conjunto de treinamento convirja para um valor mínimo. É uma boa prática fazer com que a ordem de apresentação das amostras seja feita aleatoriamente de uma época para a outra. Esta aleatoriedade tende a fazer com que a busca no espaço de pesos tenha um caráter estocástico ao longo dos ciclos de treinamento.

Para um dado conjunto de treinamento, o aprendizado por retro-propagação pode ser feito de duas maneiras básicas:

- **atualização local**: neste método, a atualização dos pesos é feita imediatamente após a apresentação de cada amostra de treinamento. Especificamente, considere uma época consistindo de *N* padrões de treinamento arbitrariamente ordenados

$\{(\mathbf{x}_1,\mathbf{s}_1),...,(\mathbf{x}_N,\mathbf{s}_N)\}$. A primeira amostra $(\mathbf{x}_1,\mathbf{s}_1)$ da época é apresentada à rede, e são efetuados o passo positivo e a retro-propagação do erro, resultando em um ajuste dos pesos das conexões. Em seguida, a próxima amostra $(\mathbf{x}_2,\mathbf{s}_2)$ é apresentada à rede e o passo positivo e a retro-propagação do erro são efetuados, resultando em mais um ajuste do conjunto de pesos. Este processo é repetido até que a última amostra seja apresentada à rede. Este método também é conhecido como método de atualização *on-line* ou *padrão a padrão*.

- **atualização em lote**: no método em lote, a atualização dos pesos só é feita após a apresentação de todas as amostras de treinamento que constituem uma época. O ajuste relativo a cada apresentação de um par de entrada é acumulado. Este método também é conhecido como método de atualização *off-line* ou *batch*.

Do ponto de vista operacional, o método local de treinamento é preferível, pois requer um menor armazenamento local para cada conexão. Como as amostras são apresentadas aleatoriamente, o uso da atualização padrão por padrão torna a busca no espaço de conexões estocástica por natureza, reduzindo a possibilidade do algoritmo ficar preso em um mínimo local. Por outro lado, a utilização do método em lote fornece uma estimativa mais precisa do vetor gradiente. Numa análise final, entretanto, a eficiência dos dois métodos depende do problema a ser tratado (HAYKIN, 1994). Neste trabalho, todas as redes implementadas utilizam o método de atualização em lote, salvo especificações em contrário.

2.4.5 Critérios de Parada

O processo de minimização do MSE (ou da função custo), em geral, não tem convergência garantida e não possui um critério de parada bem definido. Um critério de parada não recomendável, por não levar em conta o estado do processo iterativo de treinamento, é interromper o treinamento após um número fixo (definido previamente) de iterações. Serão discutidos aqui critérios de parada que levam em conta alguma informação a respeito do estado do processo iterativo, cada qual com seu mérito prático. Para formular tal critério, deve-se considerar a possibilidade de existência de mínimos locais. Seja θ^* o vetor de parâmetros (pesos) que denota um ponto de mínimo, seja ele local ou global. Uma condição necessária para que θ^* seja um mínimo, é que o vetor gradiente $\nabla J(\theta)$ (ou seja, a derivada parcial de primeira ordem) da superfície de erro em relação ao vetor de pesos θ seja zero em $\theta = \theta^*$. Sendo assim, é possível formular critérios de convergência (ou parada) como segue:

- *é considerado que o algoritmo de retro-propagação convergiu quando a norma Euclideana da estimativa do vetor gradiente* ($||\nabla J(\theta)||$) *atingiu um valor suficientemente pequeno.*

O problema deste critério de parada é que, para simulações bem sucedidas, o tempo de treinamento pode ser muito longo. Este critério também requer o cálculo da norma do vetor gradiente. O Capítulo 3 fará um estudo sobre técnicas de aceleração da convergência deste algoritmo.

Outra propriedade única de um mínimo, e que pode ser utilizada como critério de parada, é o fato de que a função custo, ou medida de erro, $J_{med}(\theta)$ é estacionária no ponto $\theta = \theta^*$. Assim, outro critério de parada pode ser sugerido:

- *é considerado que o algoritmo de retro-propagação convergiu quando a variação do erro quadrático de uma época para a outra atingir um valor suficientemente pequeno.*

Uma variação deste critério de parada é fazer com que o valor do erro quadrático médio $J_{med}(\theta)$ seja igual ou menor do que um limiar pré-especificado:

- *é considerado que o algoritmo de retro-propagação convergiu quando o erro quadrático médio atingir um valor suficientemente pequeno, ou seja,* $J_{med}(\theta) \leq \varepsilon$, *onde* ε *é um valor suficientemente pequeno*.

Se o critério de parada, por exemplo, é um valor mínimo para o MSE então não podemos garantir que o algoritmo será capaz de atingir o valor desejado. Por outro lado, ao tomarmos como critério de parada um valor mínimo para a norma do vetor gradiente, então devemos estar conscientes de que o algoritmo, provavelmente, irá produzir como resultado o mínimo local mais próximo da condição inicial. A Figura 2.13 ilustra uma superfície de erro, e apresenta o possível comportamento do algoritmo. Neste exemplo, se o critério de parada fosse $J_{med}(\theta) \leq \varepsilon$, então o método não seria capaz de encontrá-lo, pois o valor mínimo da superfície de erro é maior do que o valor de ε desejado (linha tracejada).

Atualmente existe uma grande quantidade de pesquisadores procurando funções de erro (custo) alternativas, com o objetivo de melhorar as características de convergência dos perceptrons de múltiplas camadas. Uma abordagem possível é adotar um funcional de erro que aumente a capacidade da rede escapar de mínimos locais (comentários sobre a superfície de erro e mínimos locais serão feitos posteriormente).

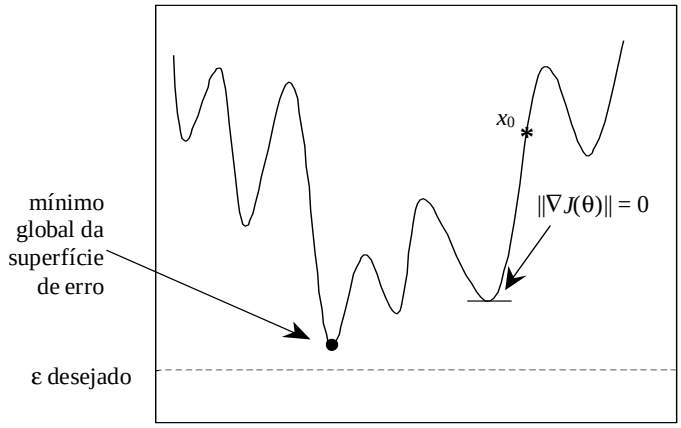


Figura 2.13: Superfície de erro com vários mínimos locais. Todos os vales (joelhos) da curva constituem mínimos locais com $\|\nabla J(\theta)\| = 0$. Partindo do ponto x_0, e utilizando como critério de parada $J_{med}(\theta) \leq \varepsilon$, o algoritmo não será capaz de determinar um conjunto de pesos capaz de satisfazê-lo independente dos mínimos locais.

Um exemplo disso é a função de custo chamada de *entropia cruzada* (*cross-entropy*) (SHEPHERD, 1997), dada por:

$$J = -\sum_{n=1}^{N}\sum_{i=1}^{S^m} \ln\left[\left(y_{i,n}^{M}\right)^{s_{i,n}} \left(1 - y_{i,n}^{M}\right)^{1-s_{i,n}} \right], \tag{2.34}$$

onde a simbologia utilizada aqui é a mesma adotada na Seção 2.4, ou seja, N é o número de amsotras, M é o número de camadas, S^m é a dimensão de cada camada e y são as saídas da rede.

Métodos híbridos que combinam diferentes critérios de parada podem ser utilizados.

Outro critério de parada bastante útil, e geralmente utilizado em conjunto com algum dos critérios anteriores, é a avaliação da capacidade de generalização da rede após cada época de treinamento. O processo de treinamento é interrompido antes que a capacidade de generalização da rede seja deteriorada. Maiores detalhes sobre generalização e procedimentos de parada utilizando critérios que consideram o desempenho da rede após o treinamento serão estudados no Capítulo 6.

Neste trabalho, utilizou-se como critério de parada o último apresentado, ou seja, $J_{med}(\theta) \leq \varepsilon$, salvo especificação em contrário.

2.4.6 A Capacidade de Aproximação Universal

Uma arquitetura do tipo MLP pode ser vista como uma ferramenta prática geral para fazer um *mapeamento não-linear de entrada-saída*. Especificamente, seja k o número de entradas da rede e m o número de saídas. A relação entrada-saída da rede define um mapeamento de um espaço euclidiano de entrada k-dimensional para um espaço euclidiano de saída m-dimensional, que é infinitamente continuamente diferenciável.

CYBENKO (1989) foi o primeiro pesquisador a demonstrar rigorosamente que uma rede MLP com uma única camada intermediária é suficiente para aproximar uniformemente qualquer função contínua que encaixe em um hipercubo unitário.

O *teorema da aproximação universal* aplicável a redes MLP é descrito abaixo:

Teorema: *Seja $f(.)$ uma função contínua não-constante, limitada, e monotonicamente crescente. Seja $\boldsymbol{I}_p$ um hipercubo unitário p-dimensional $(0,1)^p$. O espaço das funções contínuas em $\boldsymbol{I}_p$ é denominado $\boldsymbol{C}(\boldsymbol{I}_p)$. Então, dada qualquer função $g \in \boldsymbol{C}(\boldsymbol{I}_p)$ e $\varepsilon > 0$, existe um inteiro M e conjuntos de constantes reais α_i e w_{ij}, onde $i = 1, \ldots, M$ e $j = 1, \ldots, p$, tais que pode-se definir*

$$F(x_1, x_2, ..., x_p) = \sum_{i=1}^{M} \alpha_i f\left(\sum_{j=1}^{p} w_{ij} x_j - w_{0i} \right), \tag{2.35}$$

como uma aproximação da função $g(.)$ tal que,

$$\left| F(x_1, x_2, ..., x_p) - g(x_1, x_2, ..., x_p) \right| < \varepsilon$$

Para todo $\{x_1, \ldots, x_p\} \in \boldsymbol{I}_p$.

Prova: veja CYBENKO (1989).

Este teorema é diretamente aplicável aos perceptrons de múltiplas camadas. Primeiramente percebemos que a função logística, ou tangente hiperbólica, utilizada como a não-linearidade de um neurônio é contínua, não-constante, limitada, e monotonicamente crescente; satisfazendo as condições impostas à função $f(.)$. Em seguida, verificamos que a equação (2.35) representa as saídas de uma rede MLP descrita como segue:

- a rede possui p nós de entrada e uma única camada intermediária consistindo de M unidades; o conjunto de entradas é $\{x_1, \ldots, x_p\}$
- o neurônio intermediário i possui pesos $w_{i1}, \ldots, w_{ip}$ e limiar w_{01}
- a saída da rede é uma combinação linear das saídas das unidades intermediárias, com $\alpha_1, \ldots, \alpha_M$ definindo os coeficientes dessa combinação.

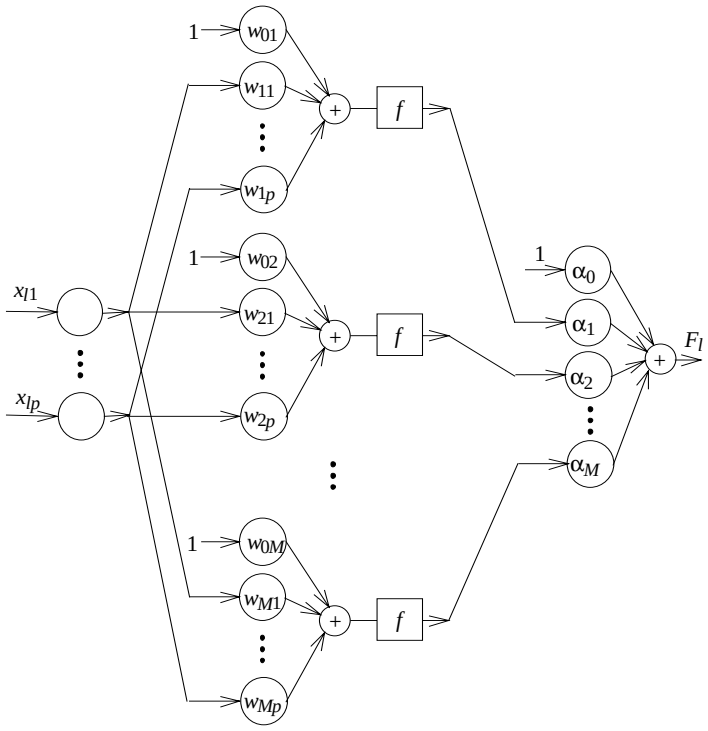
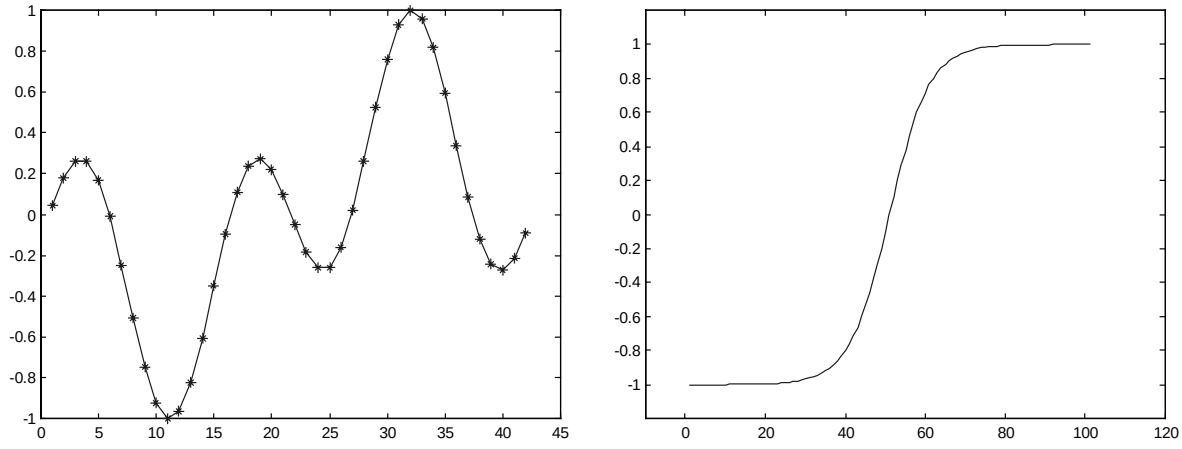


Figura 2.14: Rede MLP como aproximador universal. Os parâmetros da rede compõem a equação (2.35).

O teorema de aproximação universal é um *teorema de existência* no sentido de que fornece uma justificativa matemática para a aproximação de uma função contínua arbitrária, em oposição à representação exata. A equação (2.35) simplesmente generaliza aproximações por uma série de Fourier finita. O teorema afirma que *um perceptron de múltiplas camadas com uma única camada intermediária é capaz de realizar uma aproximação uniforme, dado um conjunto de treinamento suficientemente significativo para representar a função*. Por outro lado, o teorema não afirma que um MLP com uma única camada é ótimo no sentido de tempo de processamento, facilidade de implementação e eficiência na representação.

Exemplo 2.1:

Para ilustrar a capacidade de aproximação universal das redes do tipo MLP, considere o seguinte exemplo: desejamos aproximar um período da função sen(x)×cos(2x) utilizando uma composição aditiva de funções básicas (f será tomado como a tangente hiperbólica) sob a forma da equação (2.35). A Figura 2.15 ilustra o problema.



(a) (b)

Figura 2.15: Aproximação universal. (a) Função a ser aproximada (informação disponível: 42 amostras do valor da função em pontos igualmente espaçados do domínio da função). (b) Função básica utilizada na aproximação. Tangentes hiperbólicas, satisfazendo as condições estabelecidas pelo teorema.

A rede utilizada para aproximar esta função é uma rede do tipo MLP com uma entrada, cinco unidades intermediárias e uma saída, descrita na Figura 2.16. O algoritmo de treinamento é o de retro-propagação, apresentado anteriormente. Analisando esta figura, e considerando que a rede apresentada possui saída linear, verificamos que *y* (saída da rede após a apresentação de cada amostra de treinamento) será dada por:

$$y = w_{0,1} + w_{1,1}\, z_1 + w_{2,1}\, z_2 + w_{3,1}\, z_3 + w_{4,1}\, z_4 + w_{5,1}\, z_5, \tag{2.36}$$

onde

$$z_l = f\left(\sum_{j=1}^{k} w_{jl} x_j - w_{0l} \right), \qquad \text{para } l = 1,...,5, \tag{2.37}$$

onde $f(.)$ é a tangente hiperbólica e k é o número de entradas da rede. Esta expressão está de acordo com a equação (2.35) do teorema da aproximação universal das redes MLP.

Após treinar esta rede e definir um conjunto de pesos, vamos fazer uma análise gráfica dos componentes da equação (2.36) que determinam a saída da rede.

A Figura 2.17 apresenta as funções de ativação de cada unidade intermediária ao final do processo de treinamento. Percebe-se que as curvas apresentadas são as funções básicas (ver Figura 2.15(b)) deslocadas em relação à abcissa e devidamente escalonadas.

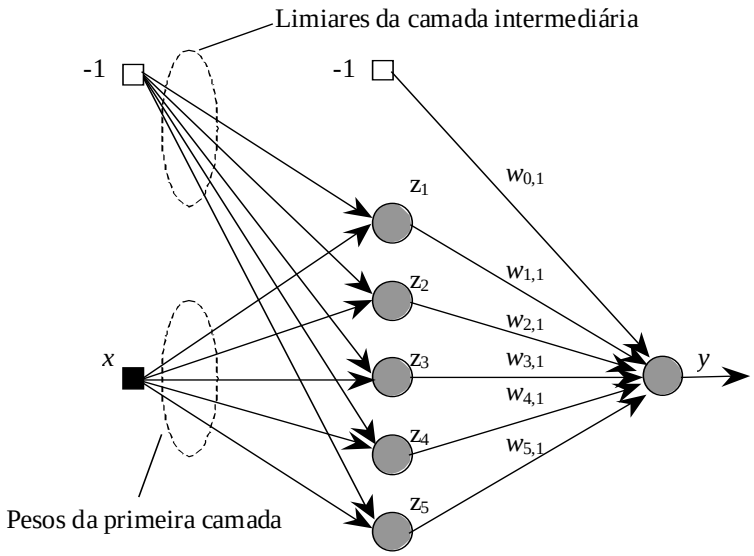


Figura 2.16: Rede MLP com treinamento via retro-propagação utilizada para aproximar a função sen(x)×cos(2x). Funções de ativação do tipo tangente hiperbólica.

Na Figura 2.18 temos, em cada caso, a representação da diferença entre a saída desejada e a saída de cada unidade intermediária ponderada pelo respectivo peso da camada de saída. Cada figura apresenta a saída desejada menos a soma das saídas ponderadas das demais unidades.

A Figura 2.19 apresenta a combinação linear das saídas das unidades intermediárias, onde os coeficientes da combinação são os pesos da segunda camada da rede.

Finalmente, a Figura 2.20 apresenta a aproximação obtida pela rede MLP após o processo de treinamento.

Verifica-se que a arquitetura do tipo perceptron com uma única camada intermediária, e cinco neurônios nesta camada, foi capaz de aproximar uma função arbitrária, no caso sen(x)×cos(2x), partindo de um conjunto de 42 amostras de treinamento.

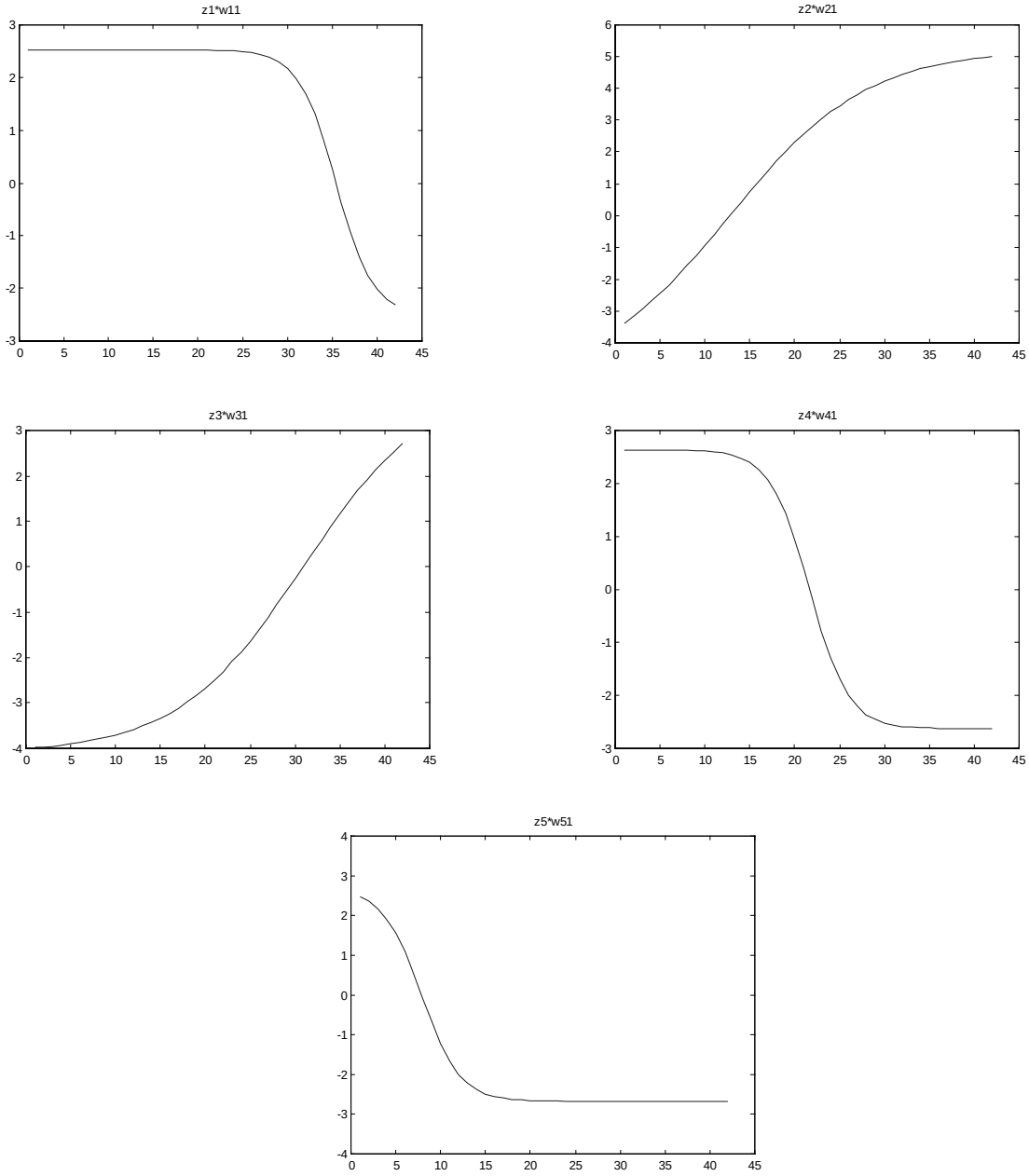


Figura 2.17: Funções de ativação dos neurônios intermediários após o treinamento da rede multiplicadas pelos pesos correspondentes da camada de saída. É perceptível que todas as funções apresentadas são tangentes hiperbólicas (funções básicas), mas estão deslocadas em relação à abcissa.

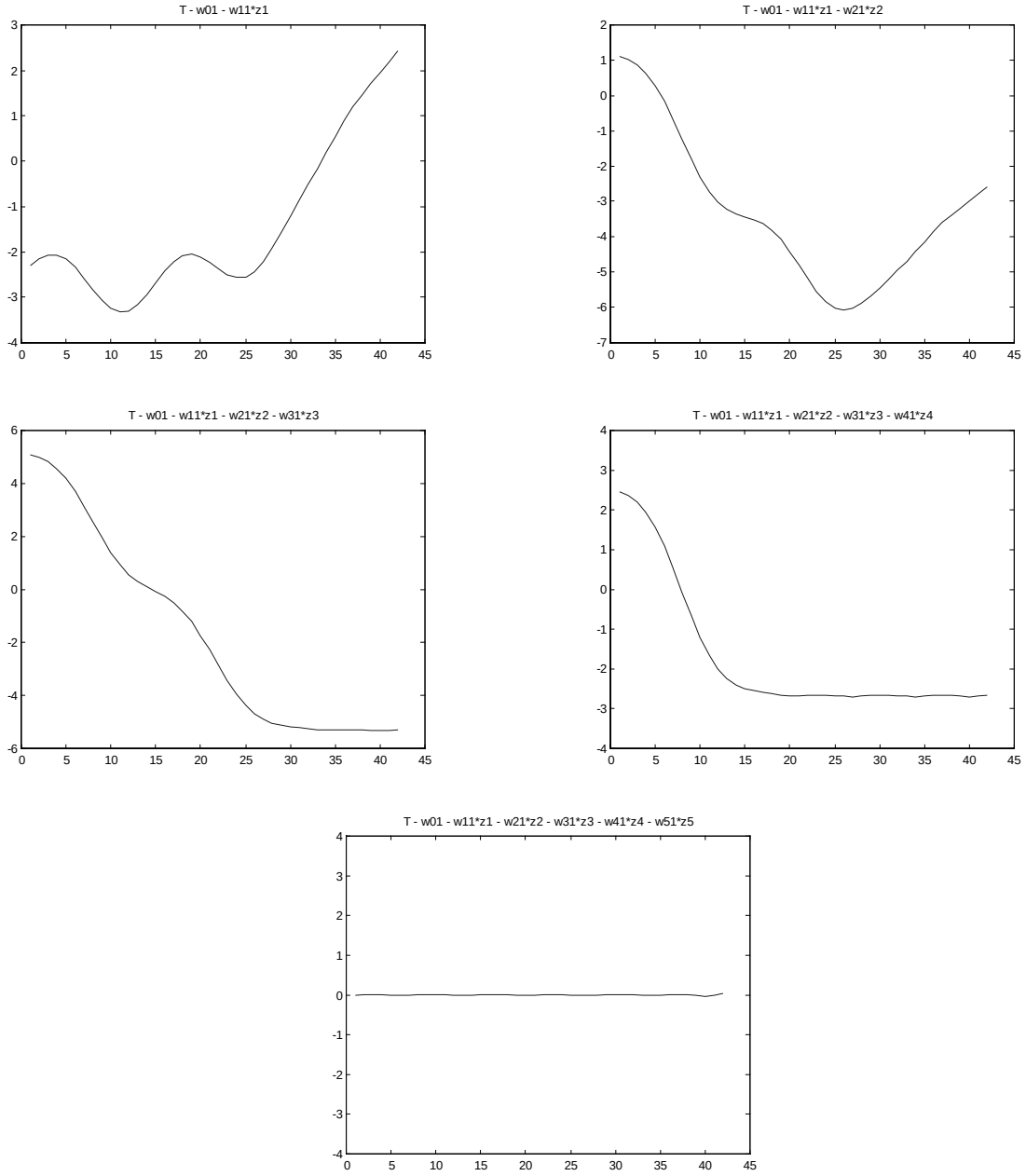


Figura 2.18: Saída desejada menos a soma da contribuição de cada neurônio da camada intermediária. O resultado final (última curva) é a diferença entre a saída desejada e a saída da rede, ou seja, a combinação linear das saídas das unidades intermediárias. Como era de se esperar, ao final do treinamento, a diferença entre a saída fornecida pela rede e a saída desejada (função a ser aproximada) é aproximadamente zero.

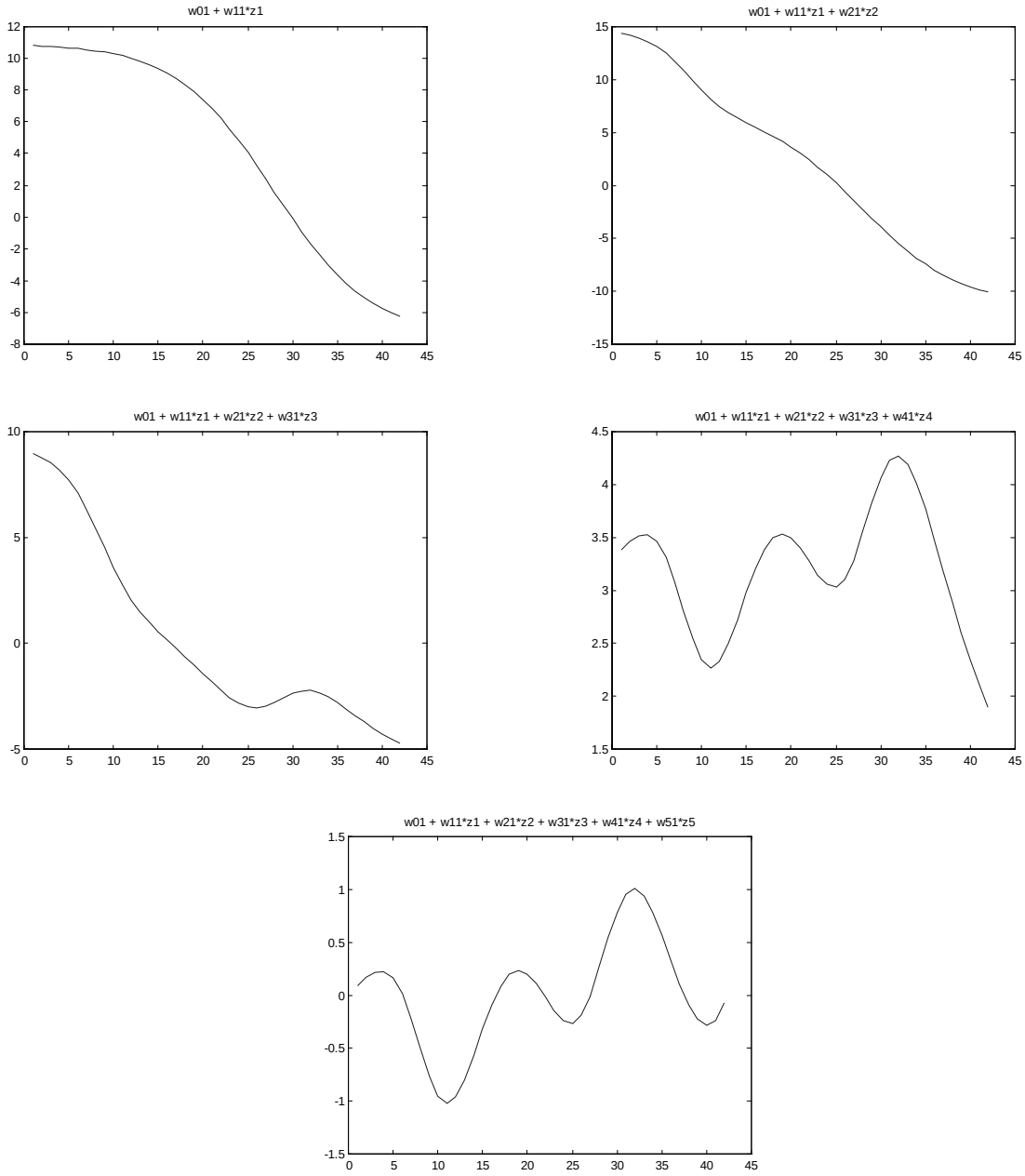


Figura 2.19: Combinação linear das saídas das unidades intermediárias. Os coeficientes da combinação são os pesos da segunda camada. Esta figura apresenta a soma das curvas que compõem a Figura 2.17, implicando na composição aditiva (combinação linear) das funções básicas dos neurônios da rede.

A saída da rede também pode ser vista na equação (2.36).

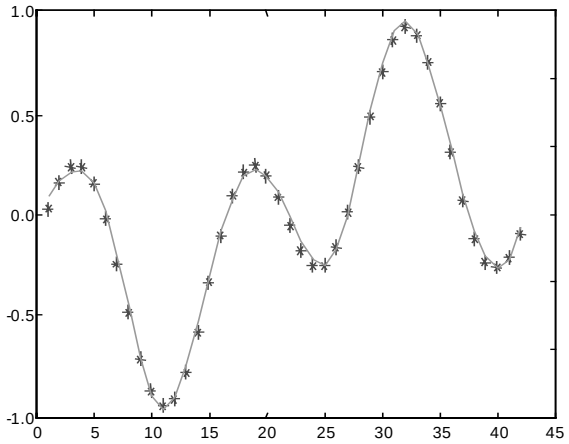


Figura 2.20: Aproximação obtida pela rede MLP para a função sen(x)×cos(2x). Os *'s são as amostras de treinamento, e o traço contínuo (—) a aproximação da rede.

2.4.7 Virtudes e Limitações das Redes MLP

O algoritmo de retro-propagação tornou-se o algoritmo mais popular para o treinamento supervisionado do perceptron de múltiplas camadas. Basicamente, é uma composição de um método de obtenção do vetor gradiente e de uma técnica de otimização (*steepest descent*). O algoritmo de retro-propagação possui duas propriedades distintas:

- é simples de calcular localmente e
- realiza um gradiente descendente no espaço de conexões (pesos).

Estas duas propriedades do método de retro-propagação são responsáveis pelas suas vantagens e desvantagens descritas a seguir.

2.4.7.1 Conexionismo

O algoritmo de retro-propagação é um exemplo de paradigma conexionista que utiliza cálculos locais para determinar a capacidade de processamento de informações das redes neurais artificiais. A utilização de cálculos locais no projeto de arquiteturas neurais artificiais, é geralmente desejada por três razões:

- as redes neurais artificiais são vistas como metáforas para as redes biológicas;
- os cálculos locais favorecem a utilização de arquiteturas paralelas como um método eficiente para a implementação das RNA's.

Comentários sobre a implementação em paralelo de redes do tipo MLP serão feitos no Capítulo 7.

2.4.7.2 Unidades Intermediárias

As unidades intermediárias exercem um papel crítico na operação das redes MLP pois funcionam como *detetores de características* (HAYKIN, 1994). Uma nova forma de exploração deste importante atributo é no reconhecimento de características significativas que identificam os padrões de entrada de interesse.

Uma rede MLP pode operar de forma auto-organizada, e quando isso ocorre é denominado de sistema *autocodificador*. Basicamente, essa estrutura faz uma *análise de componentes principais* dos dados de entrada, fornecendo uma ferramenta ótima para a redução da dimensionalidade (compressão dos dados) do conjunto de entrada.

2.4.7.3 Superfícies de Erro e Mínimos Locais

Para o desenvolvimento de algoritmos de treinamento eficientes, é essencial a compreensão das características principais dos aspectos envolvidos no treinamento das redes MLP e suas implicações. Uma maneira extremamente útil de descrever um problema de treinamento é sob a forma de *aproximação de funções*, ou *otimização de funções* – que implica na minimização do funcional de erro *J*. Sob este ponto de vista, o treinamento de redes MLP é um exemplo de procedimento de *minimização do erro* ou *otimização*, e cada problema de treinamento define uma *superfície de erro* (ou de *energia*) multi-dimensional não negativa, que é formada plotando-se o valor do erro *J* para todos os parâmetros do vetor de pesos (θ) da rede.

Esta abordagem para o treinamento de redes MLP traz vários benefícios importantes:

- muitas estratégias eficientes de otimização de funções têm sido desenvolvidas fora da área de redes neurais artificiais. Como veremos no próximo capítulo, muitas dessas estratégias são diretamente aplicáveis ao treinamento de redes MLP;
- o conceito de superfície de erro da rede MLP permite uma visualização do processo de treinamento, pelo menos quando o número de parâmetros é reduzido.

Para cada combinação possível de arquitetura MLP, problema de aplicação e função de erro, existe uma superfície de erro correspondente de dimensão $P + 1$, para um MLP com P pesos (SHEPHERD, 1997), sendo impraticável produzir um mapeamento de superfície de erro que seja detalhado e abrangente (mesmo para pequenos problemas e redes de dimensão reduzidas). Não é surpresa, portanto, que as propriedades das superfícies de erro das redes MLP sejam assunto de grandes debates.

É importante mencionar que a prática comum de inferência de características presentes na superfície de erro é bastante questionável. Por exemplo, há uma tendência de afirmar que uma rede MLP está presa em um mínimo local sempre que a curva de decaimento do erro (gerada plotando-se o valor do erro ao longo de várias épocas de treinamento) possui um grande platô em um valor de erro elevado. Na verdade, existem várias causas igualmente plausíveis para este comportamento que não são relacionadas com a presença de mínimos locais; por exemplo, a rede está convergindo para um *ponto de sela*; o problema possui um erro final elevado na solução e a rede está convergindo para o mínimo global; a rede está atravessando um platô lentamente; o algoritmo está fazendo um zig-zag no fundo de um vale; um ou mais neurônios da rede estão saturados ou a rede está tomando passos muito pequenos a cada iteração.

Algumas evidências (como gráficos de contorno de pequenas regiões de superfícies de erro) sugerem que a maior parte das superfícies de erro das redes MLP compartilham várias características (HAGAN *et. al*, 1997):

- alto grau de suavidade;
- grandes platôs;
- pequenos vales;
- vários mínimos locais e
- um certo grau de simetria em relação a origem do sistema de coordenadas utilizado para plotar a superfície de erro.

Estas características das superfícies de erro fornecem boas indicações de quais estratégias devem ser adotadas no desenvolvimento de métodos práticos e eficientes de treinamento:

- a suavidade das superfícies de erro sugerem que métodos de otimização clássica (ver Capítulo 3) que utilizam derivadas serão eficientes;
- o efeito da precisão de ponto-flutuante torna-se um aspecto importante em regiões de platôs (ver Capítulo 5) e
- se existem mínimos locais, estratégias de busca do mínimo global, ou mínimo local satisfatório, devem ser empregados (ver Capítulo 7).

A seguir apresentaremos algumas figuras com o objetivo de ilustrar as características das superfícies de erro e seus mínimos.

Exemplo 2.2:

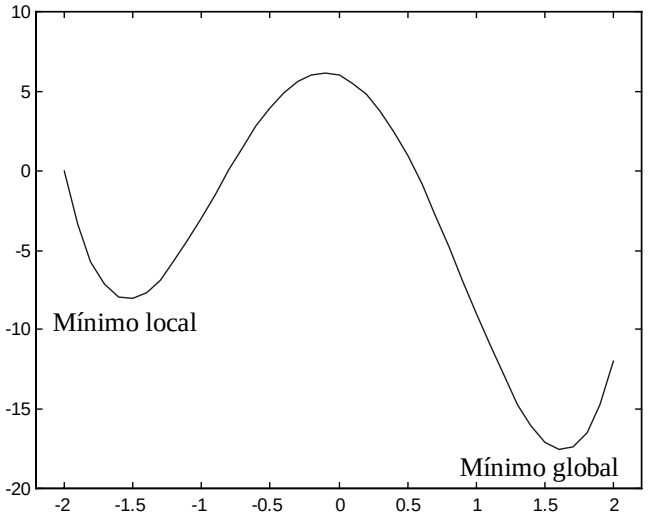


Figura 2.21: Exemplo escalar de uma função com um mínimo local e seu mínimo global.

A Figura 2.21 apresenta a função escalar dada pela expressão $F(x) = 3x^4 - 15x^2 - 3x + 6$, que possui um mínimo local e seu mínimo global para $x \in [-2,2]$.

Uma forma simples de reduzir as chances de se ficar preso em um mínimo local é escolhendo um conjunto de pesos iniciais ótimo. No Capítulo 7 serão feitas considerações sobre diferentes procedimentos de inicialização do processo de treinamento que permitem a determinação de mínimos locais mais adequados, ou até mesmo do mínimo global.

Exemplo 2.3:

Para investigar o comportamento da superfície do erro quadrático médio para redes com múltiplas camadas adotaremos um exemplo simples de aproximação de funções. A rede utilizada neste problema será apresentada na Figura 2.22.

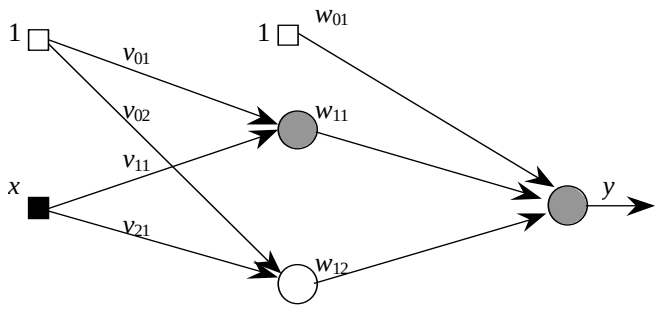


Figura 2.22: Rede para aproximação de funções.

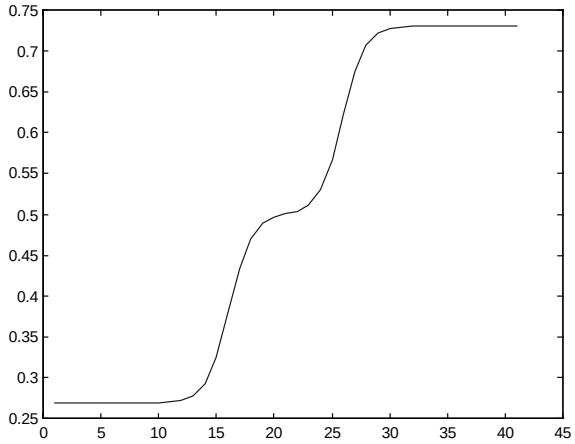


Figura 2.23: Função a ser aproximada.

A Figura 2.23 apresenta o gráfico da função a ser aproximada (HAGAN *et. al*, 1997). O objetivo é treinar a rede da Figura 2.22 para aproximar a função da Figura 2.23.

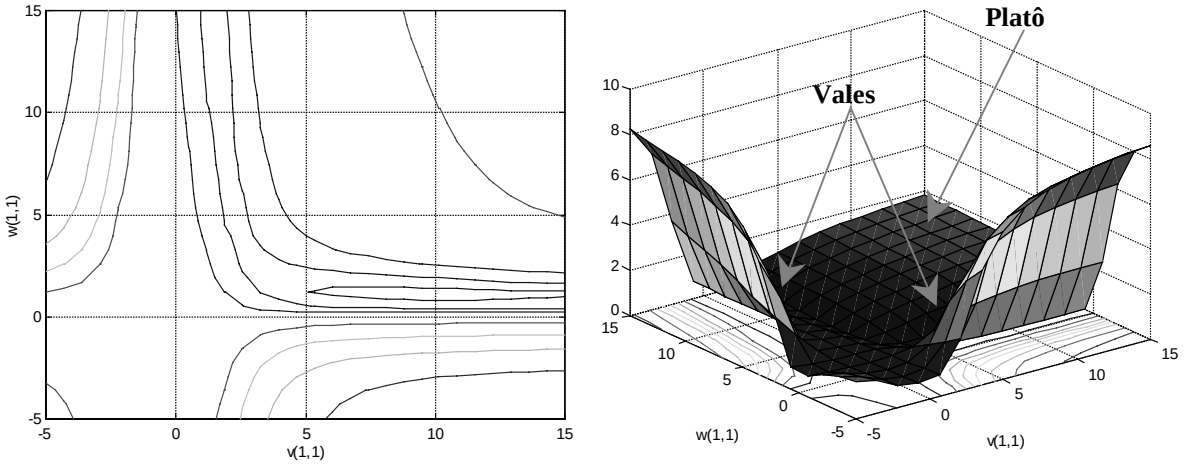


Figura 2.24: Superfície do erro quadrático e seu contorno em relação aos pesos v_{11} e w_{11}.

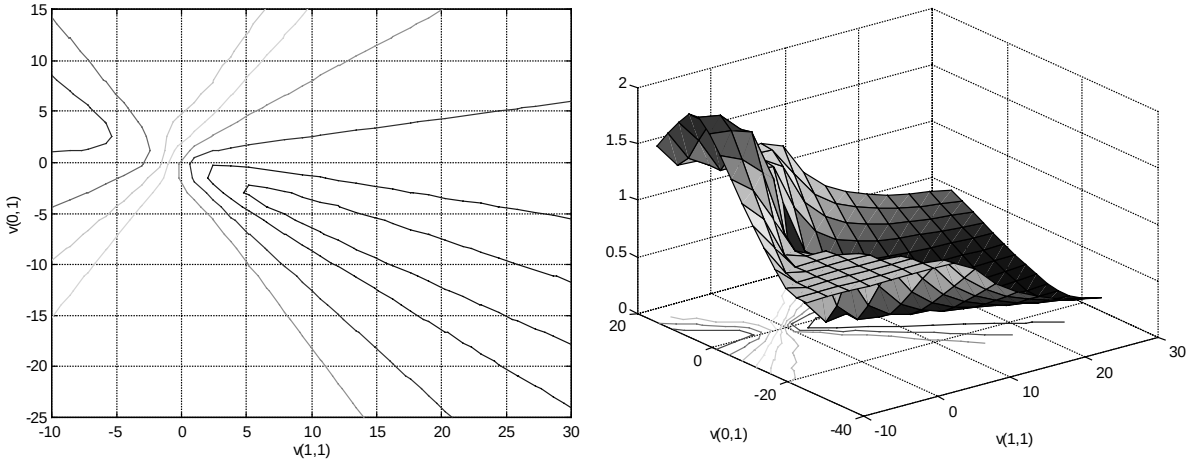


Figura 2.25: Superfície do erro e seu contorno em relação ao peso v_{11} e ao limiar v_{01}.

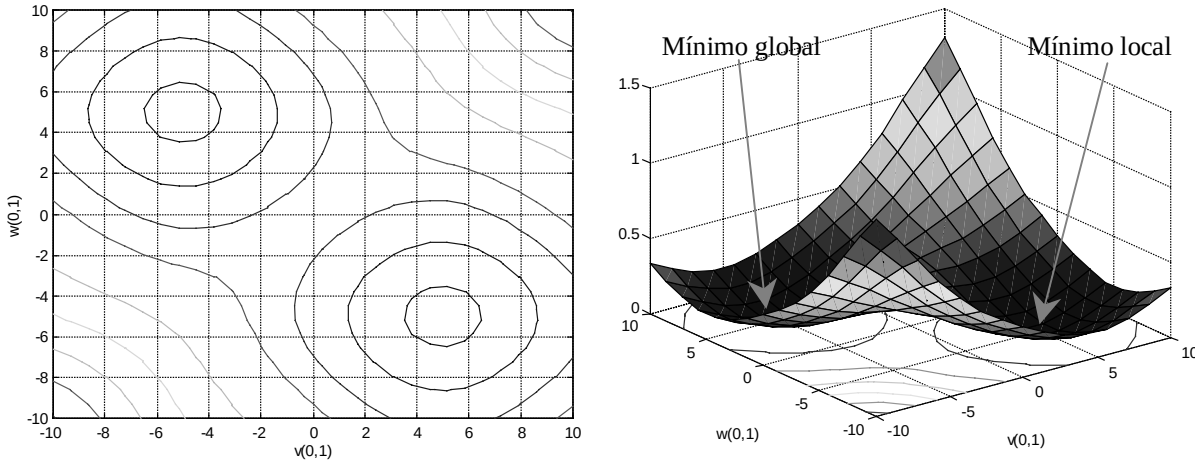


Figura 2.26: Superfície do erro quadrático e seu contorno em relação aos limiares v_{01} e w_{01}.

Para que seja possível plotar o gráfico do comportamento do funcional de erro, iremos variar somente dois parâmetros da rede a cada instante. As Figuras de 2.24 a 2.26 mostram a superfície de erro e seu respectivo contorno quando consideramos a sua variação em relação a alguns pares de pesos da rede.

As partes mais escuras das superfícies representam vales, ou regiões de mínimos. As curvas apresentadas permitem verificar algumas características importantes da superfícies de erro, como:

- suavidade;
- existência de platôs e vales; e
- presença de mínimos locais.

Fica claro na Figura 2.26 que a escolha inadequada de uma condição inicial para o algoritmo de treinamento pode fazer com que o algoritmo convirja para um ponto sub-ótimo. Como, geralmente, não conhecemos a superfície de erro, torna-se necessária a utilização de várias condições iniciais diferentes para aumentar as chances do algoritmo convergir para o mínimo global. Maiores comentários sobre condições iniciais ótimas do processo de treinamento serão feitas no Capítulo 7.

2.4.7.4 Escalonamento

Em princípio, redes neurais como as do tipo MLP treinadas com o algoritmo de retro-propagação oferecem um grande potencial. Entretanto, para que este potencial possa ser plenamente atingido, é preciso superar o *problema de escalonamento*, que refere-se à qualidade do comportamento da rede (como o tempo de processamento e capacidade de generalização) quando a tarefa a ser realizada aumenta em dimensão e complexidade. Existem

várias formas possíveis de medir a dimensão e complexidade de um problema. O Capítulo 5 apresenta algumas ferramentas que permitem fazer este tipo de medida baseado no conjunto de dados disponível para o treinamento.

Uma forma de amenizar problemas de escalonamento é fazer com que a arquitetura da rede e as restrições impostas ao conjunto de pesos incorporem informações sobre o problema que deverá ser resolvido. Outra forma de tratar o problema de escalonamento é reformular o processo de aprendizagem inserindo *modularidade* na arquitetura da rede. Um sistema computacional é dito ser *modular* se a arquitetura puder ser separada em dois ou mais pedaços responsáveis por partes distintas e possivelmente mais simples do conjunto de entradas.

2.5 Representação do Conhecimento em Redes Neurais Artificiais

As redes neurais artificiais utilizam a estrutura de interconexões para representar o conhecimento. Isso significa que, com base na configuração resultante da rede, a intensidade das conexões é ajustada com o objetivo de armazenar informação. Como resultado natural deste processo de ajuste, o conhecimento armazenado fica distribuído pela rede neural. Esta é uma das mais importantes diferenças entre processadores de informação baseados em redes neurais e aqueles baseados nos princípios da máquina de VON NEUMANN (1958), como no caso dos computadores digitais, nos quais o conhecimento é particionado em células de informação (unidades mais simples de conhecimento) que possam ser armazenadas em posições de memória bem definidas. É importante observar que todo conhecimento passível de representação em máquinas de von Neumann pode também ser adequadamente representado em redes neurais artificiais, por exemplo, tomando-se um conjunto de pesos onde as células de informação podem ser armazenadas. Portanto, o tipo de representação de conhecimento empregado no caso das redes neurais artificiais fornece a este sistemas potencialidades de representação ausentes em outros sistemas tradicionais de representação do conhecimento.

Considerando agora que todo conhecimento pode ser representado como parâmetros de um problema, a capacidade de representação das redes neurais artificiais geralmente excede a dimensão paramétrica do problema. Com isso, a relação existente entre os pesos da rede neural e as células de informação a serem armazenadas não é direta. Diz-se então que o conhecimento está distribuído pelas conexões da rede, sendo que cada conexão pode participar na representação de mais de uma célula de informação. Ao contrário do caso de máquinas de von Neumann, onde a perda ou falha de uma única posição de memória pode

acarretar a falha de todo o sistema, em representações baseadas em redes neurais, a sensibilidade à perda ou falha de uma conexão pode ser muito baixa.

Existe grande expectativa de que as redes neurais artificiais exercerão um importante papel na automação do processo de aquisição de conhecimento e codificação, entretanto, o problema de conhecimento em RNA's é representado em nível sub-simbólico tornando-se difícil para a compreensão humana. Uma forma de entender o comportamento das RNA's é extraindo seu conhecimento em função de regras (HUANG & ENDSLEY, 1997).

Em um trabalho recente, GUDWIN (1996) propõe a elaboração de uma taxonomia dos tipos de conhecimento, baseada na teoria da semiótica. Os tipos elementares de conhecimento podem ser divididos em:

- remático;
- dicente e
- argumentativo.

No conhecimento argumentativo, tem-se a idéia de argumento, que corresponde a um agente de transformação de conhecimento. Um argumento tipicamente transforma um conjunto de conhecimentos, chamados de premissas do argumento, em um novo conhecimento, chamado de sua conclusão. Esta transformação é realizada por meio de uma função de transformação, chamada de função argumentativa, que caracteriza o tipo de argumento. O conhecimento argumentativo, por sua vez, é sub-dividido em conhecimento argumentativo *indutivo*, *dedutivo* ou *abdutivo*. Os conhecimentos do tipo argumentativo são considerados os mais complexos. Em linhas gerais, um argumento indutivo é aquele que modifica um conhecimento pré-existente, o argumento dedutivo permite selecionar um conhecimento dentre vários e um argumento abdutivo gera conhecimentos de uma forma qualquer.

O conhecimento remático é o tipo de conhecimento gerado pela interpretação de remas, ou termos. Esses termos são utilizados para referenciar fenômenos do ambiente, tais como experiências sensoriais, objetos, e ocorrências. Existem três tipos de conhecimentos remáticos. O conhecimento remático *simbólico, indicial* e *icônico*.

No caso do conhecimento dicente, um termo ou uma sequência de termos é utilizada para representar uma expressão, que codifica uma proposição. O que caracteriza um termo ou sequência de termos como sendo uma proposição é o fato de existir um valor-verdade associado a ele. Esse valor verdade é uma medida da crença que o sistema cognitivo tem de

que uma proposição é verdadeira. Usualmente, o valor verdade é representado por um valor entre 0 e 1. Um valor-verdade igual a 0 significa que o sistema acredita que a proposição é falsa. Um valor-verdade igual a 1 representa que o sistema acredita que a proposição é verdadeira.

Nesse contexto, é perceptível que o treinamento de redes MLP realiza um argumento do tipo indutivo, onde os pesos das conexões são modificados no intuito de realizar alguma tarefa.